MIS780 Advanced AI For Business - Assignment 2 - T2 2024

Task Number: Business Problem Name

Student Name: *enter your full name here*

Student ID: *enter your student ID here*

Table of Content

1. Executive Summary

Use this section to introduce the business problem, data set, method, experiments, and obtained results

2. Data Preprocessing

Carry out necessary data preprocessing and exploration.

```
import numpy as np
import matplotlib.pyplot as plt
import random
from tensorflow import keras
import tensorflow as tf

tf.config.list_physical_devices('GPU')

from google.colab import drive
drive.mount('/content/drive')

# to show the folders under the dataset
!ls "/content/drive/MyDrive/Colab Notebooks/Part2_WasteImages"
```

```
import os

# Set the paths to the folders containing the image files
cardboard = '/content/drive/MyDrive/Colab
Notebooks/Part2_WasteImages/Cardboard'
glass = '/content/drive/MyDrive/Colab
Notebooks/Part2_WasteImages/Glass'
metal = '/content/drive/MyDrive/Colab
Notebooks/Part2_WasteImages/Metal'
paper = '/content/drive/MyDrive/Colab
Notebooks/Part2_WasteImages/Paper'
plastic = '/content/drive/MyDrive/Colab
Notebooks/Part2_WasteImages/Plastic'
vegetation = '/content/drive/MyDrive/Colab
Notebooks/Part2_WasteImages/Vegetation'

# get a list of all files in the folder
cardboard_file_list = os.listdir(cardboard)
glass_file_list = os.listdir(glass)
metal_file_list = os.listdir(metal)
paper_file_list = os.listdir(paper)
plastic_file_list = os.listdir(plastic)
vegetation_file_list = os.listdir(vegetation)

# print the total number of files
print(f'Total number of files under cardboard folder are:
{len(cardboard_file_list)}')
print(f'Total number of files under glass folder are:
{len(glass_file_list)}')
print(f'Total number of files under metal folder are:
{len(metal_file_list)}')
print(f'Total number of files under paper folder are:
{len(paper_file_list)}')
print(f'Total number of files under plastic folder are:
{len(plastic_file_list)}')
print(f'Total number of files under vegetation folder are:
{len(vegetation_file_list)}')

import os
import tensorflow as tf

# Create a list to store the image data and labels
data_task2 = []

# Iterate through the files in the first folder
for file in os.listdir(cardboard):
  # Check if the file is a jpeg or jpg file
  if file.endswith('.jpeg') or file.endswith('.jpg'):
    # Load the image data from the file using TensorFlow
    img = tf.io.read_file(os.path.join(cardboard, file))
```

```
    img = tf.image.decode_jpeg(img,channels=3)
    img = tf.image.resize(img, (50, 50))
    # Assign a label to the file
    label = 'cardboard'
    # Add the image data and label to the data list
    data_task2.append((img, label))

# Iterate through the files in the second folder
for file in os.listdir(glass):
  # Check if the file is a jpeg or jpg file
  if file.endswith('.jpeg') or file.endswith('.jpg'):
    # Load the image data from the file using TensorFlow
    img = tf.io.read_file(os.path.join(glass, file))
    img = tf.image.decode_jpeg(img,channels=3)
    img = tf.image.resize(img, (50, 50))
    # Assign a label to the file
    label = 'glass'
    # Add the image data and label to the data list
    data_task2.append((img, label))

for file in os.listdir(metal):
  # Check if the file is a jpeg or jpg file
  if file.endswith('.jpeg') or file.endswith('.jpg'):
    # Load the image data from the file using TensorFlow
    img = tf.io.read_file(os.path.join(metal, file))
    img = tf.image.decode_jpeg(img,channels=3)
    img = tf.image.resize(img, (50, 50))
    # Assign a label to the file
    label = 'metal'
    # Add the image data and label to the data list
    data_task2.append((img, label))

for file in os.listdir(paper):
  # Check if the file is a jpeg or jpg file
  if file.endswith('.jpeg') or file.endswith('.jpg'):
    # Load the image data from the file using TensorFlow
    img = tf.io.read_file(os.path.join(paper, file))
    img = tf.image.decode_jpeg(img,channels=3)
    img = tf.image.resize(img, (50, 50))
    # Assign a label to the file
    label = 'paper'
    # Add the image data and label to the data list
    data_task2.append((img, label))

for file in os.listdir(plastic):
  # Check if the file is a jpeg or jpg file
  if file.endswith('.jpeg') or file.endswith('.jpg'):
    # Load the image data from the file using TensorFlow
    img = tf.io.read_file(os.path.join(plastic, file))
    img = tf.image.decode_jpeg(img,channels=3)
```

```
    img = tf.image.resize(img, (50, 50))
    # Assign a label to the file
    label = 'plastic'
    # Add the image data and label to the data list
    data_task2.append((img, label))

for file in os.listdir(vegetation):
  # Check if the file is a jpeg or jpg file
  if file.endswith('.jpeg') or file.endswith('.jpg'):
    # Load the image data from the file using TensorFlow
    img = tf.io.read_file(os.path.join(vegetation, file))
    img = tf.image.decode_jpeg(img,channels=3)
    img = tf.image.resize(img, (50, 50))
    # Assign a label to the file
    label = 'vegetation'
    # Add the image data and label to the data list
    data_task2.append((img, label))

# Shuffle the data and split into train/test sets
random.shuffle(data_task2)
train_data, test_data = data_task2[:int(len(data_task2) * 0.7)],
data_task2[int(len(data_task2) * 0.3):]

# Extract the image data and labels from the training data
X_train, Y_train = zip(*train_data)

# Extract the image data and labels from the testing data
X_test, Y_test = zip(*test_data)

# Convert the image data and labels into NumPy arrays
X_train = np.array(X_train)
Y_train = np.array(Y_train)
X_test = np.array(X_test)
Y_test = np.array(Y_test)
```

3. Predictive Modeling

Create and explain your models (e.g., model architecture, model parameters). Evaluate the models on the experimental data sets.

```
# change integers to 32-bit floating point numbers
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# normalize each value for each pixel for the entire vector for each
input
X_train /= 255
X_test /= 255
```

```
# print the shape of the reshaped data
print("Training matrix shape", X_train.shape)
print("Testing matrix shape", X_test.shape)

import numpy as np

print('The original format of class of the first element in the
training dataset is: ',Y_train[0], '\n')

# Tạo một mảng NumPy chứa các nhãn rác
categories = np.array(['cardboard', 'glass', 'metal', 'paper',
'plastic', 'vegetation'])

# Tạo một bản đồ từ chuỗi danh mục đến số nguyên
category_map = {
    'cardboard': 0,
    'glass': 1,
    'metal': 2,
    'paper': 3,
    'plastic': 4,
    'vegetation': 5
}

# Mã hóa các danh mục thành số nguyên
Y_train = np.array([category_map[category] for category in Y_train])
Y_test = np.array([category_map[category] for category in Y_test])

print('The unique integer mapping encoding format of the class of the
first element in the training dataset is: ', Y_train[0])

import matplotlib.pyplot as plt

# Thay đổi kích thước hình mặc định của tất cả các hình vẽ trong
chương trình
plt.rcParams['figure.figsize'] = (9, 9)

# Danh sách các nhãn tương ứng với các loại rác
labels = ['cardboard', 'glass', 'metal', 'paper', 'plastic',
'vegetation']

# Vòng lặp để hiển thị 25 hình ảnh từ tập huấn luyện
for i in range(25):
    plt.subplot(5, 5, i + 1)  # Tạo 5 hàng và 5 cột cho các hình vẽ
    plt.imshow(X_train[i], interpolation='none')  # Hiển thị hình
ảnh từ tập huấn luyện
    plt.title("{}".format(labels[int(Y_train[i])]))  # Đặt tiêu đề
cho hình ảnh dựa trên nhãn
plt.tight_layout()  # Đảm bảo các hình không bị chồng lên nhau
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv2D, Flatten
from tensorflow.keras.layers import MaxPooling2D, Activation,
BatchNormalization
from tensorflow.keras.callbacks import TensorBoard, Callback,
EarlyStopping
from tensorflow.keras.optimizers import SGD, RMSprop, Adam, Nadam
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras import regularizers

# Sô´ lớp phân loại rác: 6 (cardboard, glass, metal, paper, plastic,
vegetation)
num_classes = 6

# Xây dựng mô hình CNN
def waste_classification_model():
    model = Sequential()

    # Layer 1: Convolutional layer
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(50, 50, 3)))  # 50x50 là kích thước hình aʾnh bạn resize
    model.add(MaxPooling2D(pool_size=(2, 2)))

    # Layer 2: Convolutional layer
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    # Flatten the layers to connect to fully connected layers
    model.add(Flatten())

    # Fully connected layer
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))

    # Output layer (6 classes)
    model.add(Dense(num_classes, activation='softmax'))

    # Compile the model
    model.compile(optimizer=Adam(learning_rate=0.001),
                  loss='sparse_categorical_crossentropy',  # Dùng
sparse_categorical_crossentropy khi nhãn là sô´ nguyên
                  metrics=['accuracy'])

    model.summary()
    return model

# Khơʾi tạo mô hình
model = waste_classification_model()
```

```
# Huấ́n luyện mô hình
history = model.fit(X_train, Y_train, batch_size=32, epochs=10,
validation_split=0.2)

# Đánh giá mô hình trên tập kiểm thử
test_loss, test_acc = model.evaluate(X_test, Y_test, verbose=2)
print('\nTest accuracy:', test_acc)

# Đánh giá trên tập huấ́n luyện
train_score = model.evaluate(X_train, Y_train, verbose=0)
print('Train loss:', round(train_score[0], 4))
print('Train accuracy:', round(train_score[1], 4), '\n')

# Đánh giá trên tập kiểm thử
test_score = model.evaluate(X_test, Y_test, verbose=0)
print('Test loss:', round(test_score[0], 4))
print('Test accuracy:', round(test_score[1], 4))

from sklearn.metrics import cohen_kappa_score, confusion_matrix,
ConfusionMatrixDisplay
import matplotlib.pyplot as plt

# Dự đoán trên tập kiểm thử
y_pred = model.predict(X_test)
y_pred_multiclass = np.argmax(y_pred, axis=1)  # Chuyển dự đoán thành
nhãn (class)
y_test_multiclass = Y_test  # Nhãn thực tế́ đã được mã hóa trước đó,
không cần dùng argmax

# Cohen Kappa Score
kappa = cohen_kappa_score(y_test_multiclass, y_pred_multiclass)
print("The result of Kappa is:", round(kappa, 3))

# Confusion Matrix (Ma trận nhầm lẫn)
cm = confusion_matrix(y_test_multiclass, y_pred_multiclass)
display = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=['cardboard', 'glass', 'metal', 'paper', 'plastic',
'vegetation'])

# Vẽ biểu đồ confusion matrix
fig = plt.figure(figsize=(11, 11))
ax = fig.subplots()
display.plot(ax=ax)
plt.show()
```

4. Experiments Report

Provide a summary of experimental results, explain the meaning of your result and how your model can be used to address the related business problem.